Improvement of an Augmented Reality Bench prototype

TR96-001

Yohan Baillot
Jannick Rolland

Center for Research and Education in Optics and Lasers (CREOL)
Department of Electrical and Computer Engineering
Department of Computer Science,
4000 Central Florida Blvd
Orlando, FL 32816-2700

Artistic view of a dynamic anatomy visualization tool in use, by courtesy of Andrei State.

CONTENTS

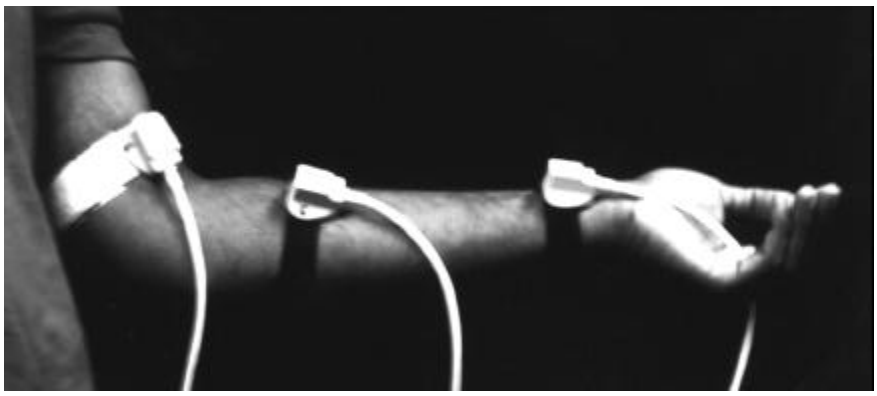
ABSTRACT

This document is the final report of the internship performed by Yohan Baillot, from February to the end of July 1996, in the Center for Research and Education in Optics and Lasers (CREOL) located at the University of Central Florida, Orlando (USA). This internship finished its Electrical Engineering major at ISIM (Montpellier, France) and followed the independent studies class performed during the preceding semester at the University of North Carolina in the department of Computer Sciences. During this period an automatic positionner controlled by a PC was designed and built for Dr.Rolland. The report “Computer-controlled positionner in depth” available in Dr.Rolland laboratory at CREOL explains more in detail how this device works.

This report deals with the adaptation of peripherals into a whole experiment dedicated to measure the ability of the human visual system to recognize the depth of virtual objects; these are displayed at theoretical depths by the way of an experimental optical see-through display. This report describes the upgrading of the positioning stage for its adaptation to the experiment, the design and implementation of a resource that make the subject able to use parallax cues, the setup of a Multi Channel Option device to transform the graphic output in two video channels NTSC, and the modification of the programmed application according the change of platform and the peripheral resources added to create the augmented reality environment.

1. DEFINITIONS OF VIRTUAL ENVIRONMENTS

A virtual environments (VE) consists of a display device that can be hold by the user, several captors and emitters to detect the position of control members or tools of the user and a graphic processing device, called scene generator, that update the display following the interaction provided by the user. This allows the user to change the display along with the motion provided in order to have the impress to inside a new world. A current example is a head mounted display (HMD) that is used to represent a virtual world moving according to the head motion of the user wearing the HMD. Virtual environments have numerous applications which use virtual worlds in order to build, explore, simulate or repair objects.



Example of “Flock of Birds”sensors fastenned to the arm of the subject to make the computer know the position of his/her elbow join.

The emerging science dealing with virtual environments is called virtual reality (VR). One branch of virtual reality is called augmented reality (AR) which is related to the virtual environments that combine virtual and real world images via a see-through head mounted display (STHMD). This tool can be used to see the inner part of a body or an object generated as a graphic image, but also the real world surrounding it, giving the impress to see through. Another application is to fit the display with tools that help the user. The high view system used for the pilot of jet to display altitude, horizon and target features on a glass to see the landscape through, was one of the first AR environments.

AR uses two different types of displays: the video and the optical STHMD. The video STHMD uses two cameras fastened on the top of the gear which take one image for each eyes of the real world, combine electronically the real images with the graphical images corresponding to the function of the VE, and display the result for each eyes. The optical STHMD used two half-reflective mirrors placed at 45 degrees to reflect toward the eyes the video outputs placed on the sides of the HMD and which display the virtual content of the image. The real world still can be seen through the optics and thus the two worlds are seen combined.

Two kind of see-through head mounted dsiplays: an optical STHMD on the left from Kaiser Electroniocs Inc, a custom video STHMD from UNC-CH. On the right.

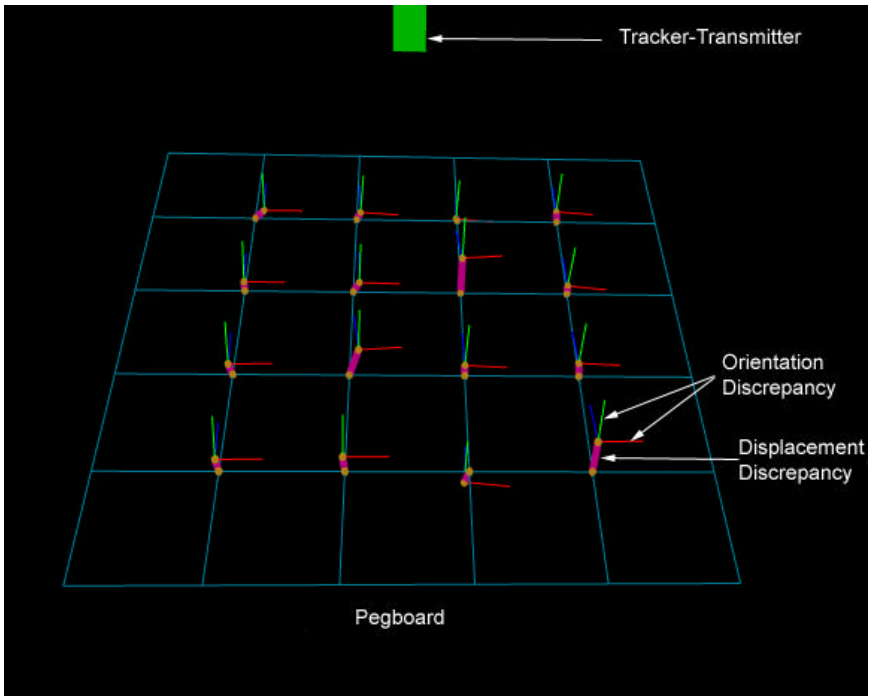
2. DESCRIPTION OF THE SETUP AND ITS APPLICATION

Optical see-through HMDs have the advantage to prevent the user from sickness reported by people in video HMD due to the lack of real clues. However, it implies to display with a higher precision the virtual world to place it correctly on the real world; with the video STHMD the resolution to reach to place the virtual world was of the pixel, but with the optical one the resolution is pratically infinite since the reality is presented directly and its resolution is infinite. Dr.Rolland started a research program in order to design and build an optical see-through tool dedicated to the teaching of radiographic positioning. This tool will display the dynamic anatomy of a subject member on top of the image of the real world (see artistic view, by Andrei State from UNC, of the tool in use on the cover page).

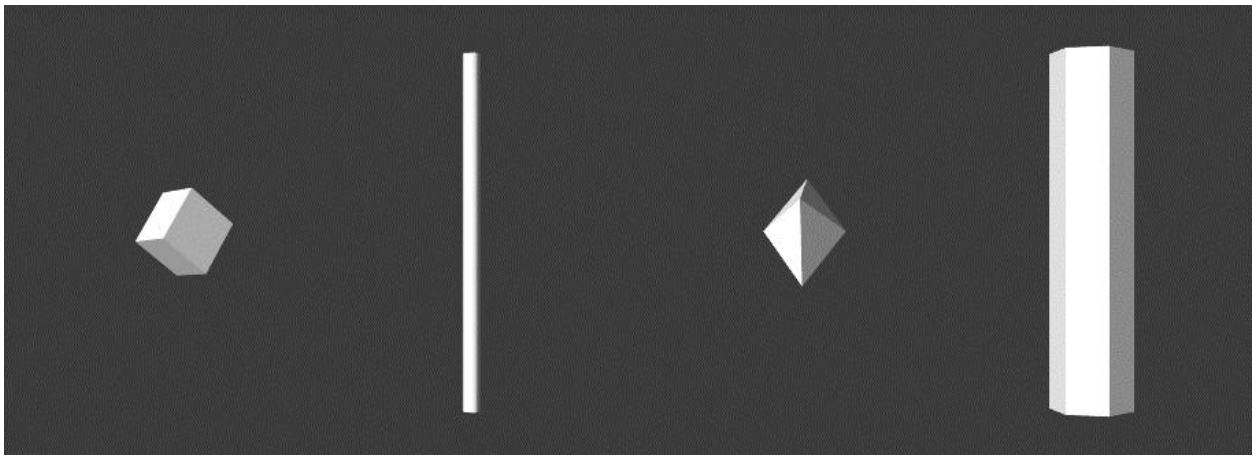
Research is currently done to determine the ability of the human visual system to recognize virtual worlds displayed using the theoretical representation models currently used. The results of this research will bring improvements to the models and will allow a more precise representation. A present experiment in this research program is the subject of this report.

The setup has been designed to measure the correctness of the model of depth representation of virtual 3D object according to the reality. In the original experiment, an real object was at a certain position and the subject was asked to judge the relative position of a virtual object placed close to but front or back to the real object. The experiment required to move the real object by hand. The new experiment will allow the application to move automatically the real object by the positioning stage. The representation of the virtual object is done via an experimental optical see-through display.

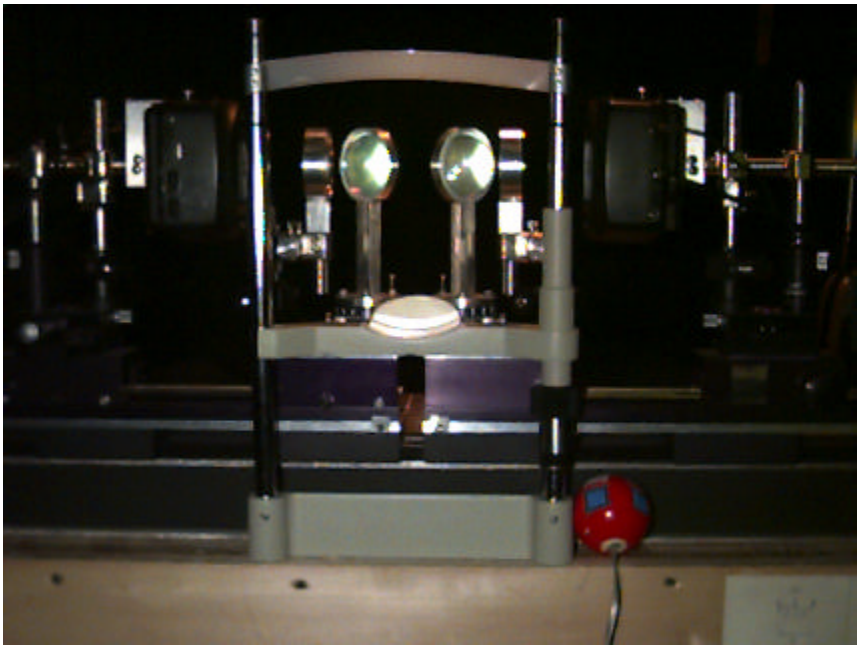
The existing setup eliminated the detection of the position and orientation of the user's head in order to measure only the errors implied by the graphical issues. In effect, from tracker to tracker, errors of tracking are often source of erroned representation of the virtual world. See below an example of the dicrepancy in orientation and position detected between the theorical plane and its registration given by a magnetical tracker.

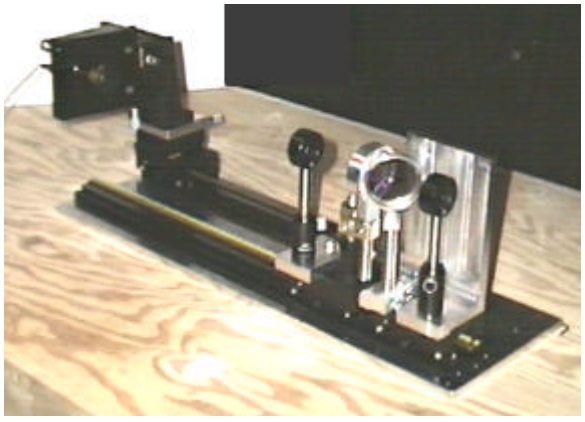


Here is an example of the virtual object stimulis used to perform the depth perception experiment.



The optical setup is composed of two arms, one for each eye; each arm is composed of a small video screen, a focus lens and a semi-silvered mirror. This two arms can be moved from each other on a precision slide in order to adapt the Inter-Pupilliary Distance of the human subject. This setup ables the user to see two different images, one for each eye. Such a system displays by this way a 3D object by stereoscopy, that means by presenting to each eye a different image to give to the subject the impress of perspective. This setup allows also to see the real world at the same time and so can be qualified of experimental STHMD with no motion possible. An illustration of the original optical setup can be seen below.


The original setup


Calibration procedure for each optical arm.

3. DESCRIPTION OF THE PROJECT OF THE INTERNSHIP

The subject of this internship was to automatize the original setup in order to improve the AR environment of the experiment. The first step was to install the positioning stage built previously in order to allow the application to move and rotate automatically the real object also. The second step was the improvement of the optical setup and the addition of a head motion parallax platform and associated peripherals. Such a platform allows the user to visualize a scene form slightly different viewpoint and provide him/her with parallax cues. This cues have been recognized to improve greatly depth perception. The third step was to configure a Multi Channel Option which is used to take the data from the frame buffer of the Onyx workstation and converts it in video NTSC signals for the video screens. Finally the original application program running on Sun workstation and Pixel Plan 5 has had to be modified in order to be run on the new platform and to take into account the new peripherals.

4. INSTALLATION OF THE ORIGINAL EXPERIMENT ON THE NEW WORKSTATION

Dr.Rolland's laboratory has been moved from the department of Computer Sciences at UNC (University of North Carolina at Chapel Hill) to CREOL (Centre for Research and Education in Optics and Lasers) at UCF (University of Central Florida at Orlando). The original experiment including the experimental optical setup and the bench table have been reinstalled. The use of high level graphics including modeling, rendering, shading, distortion correction, stereoscopic viewing and speed implies the use of a graphical workstation. The new one used is an Onyx Deskside RE2 (Reality Engine2) from Silicon Graphics Inc. (SGI) that has been upgraded with a board VTX that reduces the performance in a small amount but keep all the features of the RE2. This Onyx is equipped with one raster manager (RM) allowing to have a single graphic output on one pipe, but can be equipped with four RMs that increase the speed and the number of the graphical outputs. The raster manager is the board in charge of the rendering of the graphics in the computer. The computer runs under the control of two 150 MHz processors IP19. The I/O board is a IO4 described later. A graphic terminal of 21'' is hooked up to the server. The machine is running IRIX 5.3 operating system under UNIX. Complete characteristics of the machine can be found by using the hinv(1M) (hardware inventory) command.

The Onyx workstation has been hooked on the network and the console composed of the 21'' graphic screen and the keyboard have been connected. The basic setup have been done via graphic applications that come when the machine is booted. The intern has realized both functions of primary user and super user of this machine during the internship. The primary user is the person who use the most the machine and can benefit of special permission. The super user of a system is the person who is designated to configure the administrative parameters of the machine and tune the system in function of its use and as all the permissions for this.

These functions have been done with the help of SGI hardware support (1800 800 4SGI), the people who owned this machine before at Chapel Hill, the local support staff of the Computer Science building, the people over the Institute for Simulation and Training (IST) and several helping tools like the manual pages, release notes, Insight Library (graphic manual in-line) or newsgroups comp.sys.sgi.admin and comp.sys.sgi.hardware.

The machine has been configured as an Internet server called locally 'vision' and accessible through the network at vision.creol.ucf.edu corresponding to the resolved IP address 132.170.160.207. This has been done by modifying the /etc/hosts file system. This file has been completed by the local IP resolver servers definitions that allow the computer to resolve all IP addresses of the internet network. Two new accounts have been created in /usr/people/ directory with login 'rolland' and 'baillot' by manipulating the /etc/passwd file and setting the .login, .profile and .cshrc file in the home directory. The current window manager 4Dwm which is the default has been kept, although it will be replaced by mwm later for some accounts.

The user baillot who is the intern was also root for administration usage but the accounts were different by security. Basic free software was installed to allow to work with the system. This include the pico editor, the GNU compressor gzip/gunzip and Netscape. The home directories of Dr.Rolland and one of the user had been saved on archive 8mm tapes of 5Mb before leaving UNC by the command tar. The home directories have been installed back on vision and maya (Sparc Sun station used by Dr.Rolland) through the network and via a tape driver hooked on lorien, the mail server of the local network. Later, development software as OpenGL, CC and GCC compilers, and others libraries have been installed by mounting a distribution directory /CDROM on vision in order to use the CD player and the software available in Computer Science support center. Netscape was used to download by FTP free software like the GNU Ghost PostScript viewer.

The virtual object images were generated at UNC by the multi-computer Pixel Plane 5 which generated two outputs coming by two video cables from the computer room to the laboratory. Pixel Plane 5 is a multi-computer designed at UNC; it is a computer specialized in fast rendering of 3D structures using microprocessor-enhanced memories. A Sun workstation was used as a console to supervise the execution of the application. A version of the original application program was running on SGI Indigo for the debugging and another on Pixel Plane 5 multi-computer during the experiment itself.

The SGI Indigo version was downloaded with all the libraries non included originally on vision. This application was taken from the HMD group accessible directory ~hmd/proj/exp3d at Chapel Hill. The application was reinstalled and the executable was run successfully. The application was modified to fit more conveniently the configuration of the home directory of the intern and the new architecture. It was recompiled with the local compiler to verify the correct installation of the application and the associated libraries. However, the device used for the interactive answer of the subject was hooked up on the serial port of the Sun workstation which has a DB25 serial port and the SGI has DB9 serial port, so this part was not running during this reinstallation phase. See the section ‘interaction device’ to know how the device has been hook up again.

5. REVIEW OF INTERFACING ISSUES

The Onyx workstation from Silicon Graphics Inc. is doted of an IO4 input / output board. The board has the following ports on the front panel:

- three RS232 serial liaison ports,
- one parallel port,
- four interrupt output ports,
- two input ports,
- two video output NTSC composite,
- one network communication protocol port,
- others ports non useful for this present paper.

PARALLEL PORT

The parallel port attracting first the attention because the positioning stage was controlled via this port is unfortunately a non standard parallel port. The parallel port is a DB25 female connector (D sub connector with 25 pins). The only manual pages available on this port deal with a printer connection. The directory /dev including the files for all the devices recognized as hooked to the computer does not have a file allowing a direct control of the eight data bits by writing or reading of a word value in a register. Research through the comps.sys.sgi newsgroups, the SGI archives available on Internet and personal communications brought at the conclusion that the parallel is controlled by VMA interruption that give, by the commands available, the possibility to control a printer only. A solution was however possible on Indigo models. This solution is described next.

First trials has been done trying to open the file devices estimated as connected to the parallel port as /dev/plp (parallel printer port), /dev/cent (interface centronic), /dev/tek (Tektronix interface) and others. The uses of fopen() and fwrite(), or open() and write() have been unsuccessful.

A dummy parallel connector has been plugged in the parallel port. Oscilloscope sensors were connected to /strobe(1), /ack(10), and data0(2). /strobe (pin 1, to find pinout see man plp) is the strobe signal that give the information to the printer that the data are ready by a brief falling impulse of microseconds. The printer takes the data and acknowledge the reception by sending the similar low pulse on /ack.

The method found on Indigo to bypass the printer problem is to connect /strobe(1) to /ack(10) in order to create a feedback of the strobe signal which is then directly acknowledged. Also the /busy(11) is to draw to the ground in order to simulate a printer never busy. The control of the port is done by opening the device /dev/plp with open(2) to extract a file descriptor, write with write(2) byte wide data, and controlling parameters of the port via ioctl (C command to control I/O devices, see ioctl(2)). The ioctls particularly allowed to vary the time of the data presence but without going over 5 times the strobe during time, and also to disconnect /strobe and /ack by setting the flag to ignore the acknowledge system.

The Indigo parallel port actually latch the output of the parallel port keeping the data available until the next writing. On Onyx, the data are not latched and are only present during a period similar to the time of the strobe impulse. The solution chosen was to order a 8 D-latch buffer and an inverter in technology CMOS. Once received, these items will be soldered on a board and powered with a lab alimentation of 5 volts. The D-latch will be connected to the data pins and will be latch-controlled by the strobe signal passing through the inverter. The output of the D-latch will constitute the new data lines of the parallel port.

The reading of the parallel port can be done only on four pins that constitute the four low bits of the STATUS byte. This can be done also via ioctl calls that transfer this status byte in a system variable usable in code. Please, see codes below for examples of writing and reading. The circuit connection and its use have not been done at this time because another solution has been adopted. The inverter remaining pins will be used however and the parallel port pins will be used to add devices to the VE.

SERIAL PORT

There is four female DB9 serial ports that correspond to the device file /dev/ttyd[1-4]. These serial ports are doubled by four RS422 serial liaison ports sharing the same device files but able to provide power to the device they drive. The serial port seems to be commonly used on Onyx to communicate with virtual environment devices. This is a problem when a real-time response or registration is required because the serial liaison protocol involves a delay due to the transmission of the data with control bit and synchronization processes. More, the parallel port allows to manipulate eight bits at a time against one for the serial port.

By this port a RS232 interface could be attached and could control via timers, counters, I/O buffer and built-in controller various home made devices and decrease the CPUs load, but this solution is also very expensive. This solution has been put on the side at this time. However, the electronics box interfacing the interaction device is this kind of interface and is described in the ‘interaction device’ section. The RS422 DIN port is connected to the same logical file that the RS232 above them, its only difference is that it is powered at +12V.

OTHER CONNECTORS

The two output BNC connectors labeled ‘composite’ are connected internally to each others. They are discussed in the section ‘MCO configuration’ (see man -k video for the video command and description of the video outputs).

The interrupt ports are divided in three outputs and two inputs which represent the same logical inputs (see man ei()). The ports are controllable via ioctls calls. The interface working is explained later in this document in the section ‘Positioning stage upgrading and installation’.

The network communication /dev/et0 port has been connected to the local network T-base 10 including both the phone and computer networks. Notice that network communications are on the left RG45 plug (phone plug) the wall plug.

6. THE INTERACTION DEVICE

The answering joystick that represents the interaction device of the subject is a pool ball fitted with two single free-position switches connected to an electronic box. The electronic box is a power supply for the joystick, an analogic / digital interface and a RS232 interface. The analogic / digital interface converts the analogic signal from the joystick into digital data that can be sent by the serial Interface to the computer via the RS232 liaison.

The serial port of a Sun station was used before to interface with the electronic box. This serial port was a DB25 with 25 pins. The port of the Onyx is a DB9 with 9 pins. The use of a transducer DB25-DB9 was the first trial to be able to use the interaction system but without any success. It was found that the port of the SGI and the Sun stations have the pins 2 and 3 inverted; this 2 pins are used to transmit the data in applying a voltage difference between this two pins. A NULL-modem converter was used to try to recover this problem.

However internal connections were certainly established setting an incorrect voltage somewhere causing the system not to work. More, the superposition of both the NULL-modem and the DB9-DB25 connectors took so much space that it was not possible to close the maintenance door of the computer anymore. Then the cable original DB25 plug of the electronic box was open to figure out what were the used pins. The correspondence was done with pinout of the parallel port of the two stations (see below) found in the serial(7) manual page on SGI and on the Sun home page help on-line via Netscape. A new plug was made for the electronic box with the correct wiring.

The ‘v_setenv’ sourcable is file used before the execution of the application and in order to setup environment variables. These variables are used by the Analogical Device C library designed at UNC to control analogical devices in order to make it known what are the devices to communicate with and where they are plugged.

The port of connection where the electronic box is connected is the serial port 2 (number 1 is used pour la console) so the environment variable ‘device’ has been set to /dev/ttyd2 that correspond to this port. The connection was then successfully recovered when the application was launched.

The electronic box includes several ICs and allows probably to control other apparatus than the interaction device. The drawing of the board was purchased to a company before by UNC that reproduce it for internal use. The company has been contacted in order to receive the engineering manual. Thus, it will be possible to hook up other components via this board and may be used to modify the actual configuration in a more convenient way. Below is the just received scheme of the Little Giant microcontroller.

7. INSTALLATION AND UPGRADING OF THE POSITIONING STAGE

The positioning stage was first designed as an independent study realization. It was originally controlled by a PC. The stage gives the possibility to translate and rotate an object on a sequence programmed on the PC. It was composed of a stepper motor moving the translation stage and powered by a driver, and a DC motor with a feedback to regulate the speed provided by an infrared detector.

The experimental setup graphics generation load implies the use of a graphical workstation more than a PC. The positioning stage supervision program was written in QuickBasic and Assembly because the C language was not available on the available PC 386SX Zenith. An interpretation of the two part of the program in C has been the first step in order to be able to implement the program on the workstation. This translation has been divided in several 'modules' accomplishing each functions: read the encoder, write the driver of the stepper motor, detect the end of course switches interruptions. This division allowed to test one function at a time during the design and debugging of the code and then facilitated the design of the main application described later in 'reorganization of the application'. In the following a module has been written for each functions of the home made peripherals.

The original program was using the parallel port to send the steps frequency of the stepper motor of the translation stage and the controls signals for the rotation DC motor. The application was receiving by the same port the signals from the ends switches of the translation stage and the infrared optical encoder to regulate the rotation speed. The rotation stage was a problem with the first PC platform because of the generation of the variable duty cycle signal to rotate the DC motor at variable speed, and the reading of the encoder at the same time in order to regulate the speed which were hard to manage with only one processor. This was accomplished by programming in assembly language and emulating time sharing between the reading of the speed encoder and the sending of a control signal to the motor, in the inner itself of the assembly code.

Another restriction was that the correct control of the rotation motor was not possible when the control of the translation stage was done for the same reasons. See directory ~baillot/public/exp/old/ on vision at CREOL or on measn0-9 at ISIM to see the code employed in the old version. The two programs used were super.bas (QuickBasic) and rotate.asm (masm assembler call).

The workstation used has two microprocessors that allow to perform real-time and multiprocess programming. With the multiprocessing possibility it is possible to make turn in time-shared or even in parallel (see man sysmp()) several processes together. The section 'reorganization of the application' talks about the use of these possibilities for the final application.

The real-time capability include signal management, timing and process priority tuning. Signal can be used to synchronize one or several processes to schedule the tasks order or signal an event. Timing can be done thanks to three timers available by process: a real which is updated constantly, a virtual and a 'prof' timer that are updated following certain conditions (see man timers()). Timers can be initialized, updated and setup to send a signal SIGALARM when time expires (see getitimer manual page). The process priority control makes a process able to control its priority in order to be chosen to run at the earliest time possible or at the contrary to put itself in a background jobs category.

To control the step rhythm of the stepper the External Interrupt library (see ei() manual page) providing the user with easy control commands of the interrupt input and output ports has been used. It is in effect possible to produce impulses active low with a tunable period in the range of the stepper clock pulse width. The pulse width has been set to 20 microsecond that meet the specifications of the notice of the stepper motor driver. The interrupt output ports are high (+5 volts) by default as well as the pins of the driver.

An interruption handler has been written to catch the SIGALARM signal. The realtime timer associated to the process is initialized at the maximum period duration between two step corresponding to the slowest speed at each begin of motion. When the time is over the timer send a signal SIGALARM that stops the current execution of the process and execute the handler procedure.

This procedure has been written to make the interruptions appear at this time on the port 0 and 1 for the line /clk and /dir of the stepper motor driver. The signal clk is the stepper clock to control its speed and the dir is the signal of direction that corresponds to a move on the right of the translation stage when the pin is high (default) and a move on the left in the other case. The port 2 drives a low signal that is inverted to produce a high signal on the pin ‘ena’ (standing for enable) to let the power go through the driver when a move is performed, and keep a low state stopping to provide power to the motor when not used. This prevent the driver thermostat to stop the driver power due to a too high temperature. The only time duration of the direction and enable signal during the time of the application of the clock pulse is sufficient for the driver to produce correct steps. One of the ordered inverter remaining inputs will be used to invert the interrupt port 2 state. With this configuration the motor consume power only during the 20 microseconds of the pulse application for each step. The handler then update the timer value in order to modify the time of the next period of time before the timer sends another signal. Thus, the speed of the motor can be updated and this is controlled in a way that it produces a trapezoidal speed law whose the bases are the maximal and minimal speeds to apply.

Then the process returns to the main application repeating the last call system interrupted and following the code until the next interruption. However when the number of steps to perform has been done the process return to the main application after having send a SIGUSR1 user signal number one that can be caught to synchronize the application and turn off the timer until the next move.

The output of the end of course switches presenting a value of 0 when one of the two hall effect normally closed sensors detect the motor in an extreme position has been connected to the pin 15 of the parallel port that corresponds to the function FAULT (printer fault) handled by the flag in bit 0 of the status register which can be read via ioctl calls. A reading of this value is realized in the handler before each interrupt pulse execution and the stepper can by this way be stopped in case of problem by disabling the timer. The temperature pin could be also connected by this way but as the power is applied following the method explained before, the driver never stops due to a temperature problem.

The rotation motor static relay transforming a digital input signal of 0/5 volts in a digital copy of 0/12 volts to provide the motor with power. This control will be possible with the modification of the parallel port at the reception of the D-latch buffer. The same method of pulse width modulation has been coded in a module to control the tension applied to the DC motor and drive a signal on data pin 2 (data1) of the parallel port. However, the manual of the electronic box interfacing the interaction buttons will show if the box has timers which will be able to decrease the spent CPU time on the rotation motor by controlling the complete regulation of the speed. At the current time the module created to control the DC motor signal uses also its real-time timer that is updated with a low_time if the next state of the signal is low and vice versa. The low_time and the high_time added are a constant time value that must be as short as possible to be averaged by the motor because of its inertia; they are initialized at the beginning of a new speed consign and controlled by the feedback module. The timers send a SIGALARM signal at the end of each period calling a handler procedure similar at the one used before to update the timer and signal value. Thus, time is not spend in looping until the next period and the time CPU can be spend on the main application.

The feedback module reads the output of the infrared detector used as encoder to detect the speed and update the timer values low_time and high_time in order to vary the duty cycle of the control signal of the DC motor and thus, regulate the speed. The external interrupt input port has been used to receive the signal from the infrared reflective detector.

The detector is wired in a way described in the report ‘computer controlled positionner in depth’, but a scheme available in appendices shows the electrical connections. An ioctl call allow the user to set a signal that is sent when an interrupt arrived. The main application of the module is left when the signal is detected to execute the signal handler that measures the time since the last impulse from the encoder (see man gettimeofday()) and, after several samples have been taken, computes the actual rotation speed. The values of low_time and high_time are then updated in order to regulate the speed, sent via an interprocesses communication pipe to the control process of the DC motor. The samples number should not be too high to have a reasonable response time and not to small to compensate the imprecision of the delivery time of the signal to the signal handler.

8. A CALIBRATION PROCEDURE FOR THE POSITIONING STAGE

The translation stage did not have any feedback loop because in theory the motor is able to follow the steps correctly if the current necessary to move the load at the wanted acceleration is provided. However the lost of several steps on a course can be invisible and the result on the experiment could be wrong if this occurred. To determine the ideal acceleration without any lost (or frequency of command of the driver) an electronic <pied a coulisse> with a resolution of 0.01 mm has been employed to measure the distance between the cart and the vertical surface facing it at the extreme end of the stage from where the carriage was placed beforehand at about 10 mm. The stage was moved for each trial from left to right and right to left. The trials were begun at low acceleration to determine the correct distance and the acceleration was increased following the research by dichotomy to determine the optimal acceleration for the given load, and for the two direction of travel. A security margin of 10% has been taken from the found acceleration to be safe from the limit. Future people working on the setup should take care to redo the acceleration calibration if the load of the moving part is modified by addition of an element or modification of the structure.

This simple calibration procedure avoids the cost of a specialized feedback interface and a rotary encoder in order to perform the calibration of the acceleration automatically. A solution possible however will be to order an optical encoder adaptable on the back axis of the stepper motor and create another module that will read this feedback device. The back axis of the stepper motor can is design for this purpose and can be seen in the purchased elements documentation available in appendices. This system will allow to make automatically this calibration procedure to determine the ideal acceleration to apply to the stage, without any human intervention.

9. THE OPTICAL SETUP DESCRIPTION

Before the installation of the automatic positionner and the parallax stage, a phychophysical experiment has been conducted with the existing optical setup. The experiment was done with a real object placed at a fixed distance manually and moved from trial to trial. Three hundred trials are collected for one experiment. Two small television screens were used to display the NTSC signals issued from Pixel Plan 5. The images were focused through lenses and reflected toward the eyes by two semi-silvered beamsplitters.

The scene in front of the subject still was seen through the beamsplitters. This visualization method, explained before, allows to combine a virtual scene and a real scene and is applied for the see-through technology.

The improvement realized to this setup during the internship was the addition of the parallax resource and the change of the beamsplitters. Elliptical pellicles semi-silvered beamsplitters were used. The pellicle word refers to the fact that the active part of the beamsplitter is not a glass that measure several millimeters but a pellicle of thickness of several microns; this avoid to have a ghost image visible due to the fact that there is in fact two reflection surfaces (the front and back) in a conventional beamsplitter. The elliptical shape has been chosen to allow the conservation of a the field of view of the virtual image. The width of this round field of view was reduced because of the inclination at 45 degrees of the previous circle beamspiltter that was giving a vertical elliptic field of view. The elliptic beamsplitter has a horizontal long diameter that is dimensioned in order to have the same projected diameter than the small one when inclined at 45 degrees, giving the impress of a round aperture.

10. THE PARALLAX SETUP

The first experiment was using a fixed optical setup. As the use of parallax cues has been shown very helpful to judge the depth of an object (Rogers & Graham, 1979), the optical setup has been fitted with a resource allowing the subject to move by hand his/her head on the right and on the left along with the optical setup. It was then required that the images on the two television screens be updated with the motion of the user to visualize the correct view of the virtual object.

The original optical setup was composed of two arms holding each a television screen, a focus lens and a semi-silvered mirror. Each arm were fasten on carts movable on a graduated precision optical rail allowing adaption to the various inter-pupilliary distance (IPD) of subjects.

A rail of 1.20 m equipped of two carts has been fastened on the optical table where the original setup was placed. The original graduated slide supporting the two optical arms has been replaced by a shorter one and fasten on one side of the two carts. This continues to allow to tune precisely the inter-pupilliary distance (IPD) on a new precision slide of 0.6 m but does not increase the space in width taken by the experiment. In effect, as the parallax range that the user can cover has been fixed at 0.6 m, the space taken in width by the experiment is 1.20, the same than before. We can define an angle of rotation representing the angle that the object should be rotated to produce the same view as when the subject move the parallax on the extreme left or right. The minimal angle implied by this range is arctan(0.3/3)=5.71 degrees.

The chin-rest support has been fixed on the other side of the carts up side down. Before, the chin rest was fastened on the wood table, 5 cm lower than the optical table. As the chin rest must be on the cart like the optical setup, it would have been to high. Fortunately, the shape of the support allowed to sustain the poles of the chin rest lower than usual and avoided the design and cost of an other part. The support was fastened up side down and the two poles where by this way lowered of about the same distance that the one needed.

The design of the optical setup allowed to push up the center of the television screen and of the optics in order to catch back the remaining high, what would not have been possible if the chin-rest would have been fastened normally.

A linear encoder of 600 mm of travel was fastened next to the rail and its cart was fastened with a piece designed previously to one cart holding the optical setup. The resolution of the encoder is of 10 micrometer for the coarse resolution. The output of this linear encoder was wired to the workstation in order to communicate the position of the head of the optics and so of the point of view of the subject. The reading head of the linear encoder sliding on a graduated chrome glass scale produce TTL square signals. Two signals in quadrature of phase are cadenced following the number of graduation encountered, and another produce a low pulse on each zero-marker printed each 50 mm. These signal are called respectivelly A, B and Io. Two signals are sufficient to detect the direction of travel according the current state and the next change that occured to these bits. The two outputs A and B are read via the pin 1 and 2 of the status byte of the parallel port corresponding to pins 17 (EOI, end-of -ink) and pin 13 (ONLINE, printer is on-line) of the parallel port and accessible via the flag bits 2 and 3 of the status byte.

A driver module has been written. It reads regularly these two pins at such a rhythm that is higher that the highest change frequency implied by the maximal travel speed. This prevent the application to loose steps done by the linear encoder. The maximum speed of travel allowed with the linear encoder implies that the reading happened every 10 microseconds, what will be conserved except if it overload the genral running of the application.

The module creates a parallel process that is scheduled in a way that it is executed by the kernel during a minimal time whenever within each period of 10 microsecond with the schedctl(2) command. The process has been set in background priority in order to be executed if no other process need to run, excepted if the scheduling period intends to finish before the process had had some time to run. This process is a simple loop that polls continually the two output values from the encoder.

The minimal time allowed has been set after having measured the time spend by the process to loop one time. Each time the process is executed the new position is updated if needed. The minimal time scheduled insured that the process can at least done a reading every period but does not prevent it to continue to be scheduled if no process wait to be executed.

11. MCO CONFIGURATION

The MCO is used to produce several graphics RGB channels with each raster manager output or graphic pipe. On the Onyx that we get, the only RM output can be seen on the console. When switching in MCO mode with the superuser priority and the command written below the RM graphic output is redirected on the MCO out-break box instead of the console port of the front panel.

setmon -f format
/sys/stopgfx
/sys/startgfx

The reconfiguration of the direction of the output takes effect when the RM manager is stopped and restarted. Thus, the image on the screen disappears and there is no way to control visually the keyboard entries. There are two solution to this problem. The first is to employ a terminal, a VT100 for example, that can be plugged on the serial port one reserved for the console ASCII output via a serial communication.

The MCO grows the 1280x1024 high resolution output of the RM to an output of 2560x2048, multiplying by the fact the graphic surface by four. Several video RGB outputs are produced with parts of this surface. With one RM manager a format available is 4@640x480 at 60Hz (format VGA), what means that four outputs are produced following the dispatching given below.

This formats are convenient because close to the resolution of the television screen that is 589x435 at the most. All the formats available with the number of RMs is given in the directory /usr/gfx/ucode/RE/vs2/vof/{1rm,2rm,4rm}. The number written correspond to the channel where the output is directed on the MCO. Each output has 3 BNC connectors for the red, the green and the blue component of the graphic output, and 3 others for the Genlock, Horizontal and Vertical synchronization required with some video apparatus. As you can see on the screen division, the original top-left quart of the screen can be seen via the channel 1.

A cable was built from 3 BNC-BNC cables and a 13W3 connector which was used for the monitor and thus this parts of the screen was looped back and visible from the user. The application use this where a window of supervision is open with very small characters to catch back the scaling. The two others right parts of the division correspond to the new viewports of the graphic output for the right and left eye. This channel corresponding are respectively the 5 and 2 which have been connected to the television screens via coaxial cables and to represent the scene for the right and left eye.

12. HIGH-LEVEL GRAPHICS PROGRAMMING LEARNING

OpenGL is a C library that allow the easy programming of high-level graphics. The library is can be included like all other libraries with some #include calls in the C program. This library include primitives to compute rendering, shading, texturing and lightening. The rendering is the function to render the object, that means compute each surface position on a 2D display from its 3D coordinates, 3D view point position, the field of view and the perspective used. Some other primitive that allow culling or depth cueing can be included in rendering. Culling is a used to remove the points that are shaded by a surface between the considered point and the point of view. Depth cueing is the property that the object appear to fade when they go far away from the view point due to atmosphere and the lines seems diminish in size due to the perspective. This property give a cue to the human visual system of the distance and size in depth of an object.

As this report is not based on the knowledge of OpenGL we will limit the explanation of the used primitives of this language. A brief part of the wide domain of Computer Graphics will be explained with the primitives if needed.

Virtual reality involves two major sciences that are electrical engineering to interface the different peripherals, and computer sciences. Before this internship one semester of class has been done at the department of computer sciences at the University of North Carolina famous for its works in virtual reality. During these time the intern was studying graphics courses in order to acquire the basics skills of the high-level graphics programming that the intern did not have before. These classes emphasized C programming on workstation, algorithms for the design of drawing primitives for X workstation and modeling, as well as project of research associated to these classes to practice. Because none of these skills were known of the intern, this period along with the previous project for the design of the ramp were a key for the present internship. Later on, when this internship started OpenGL, an high-level graphics language, was learned.

OpenGL is a C library that provide 160 primitives allowing to create graphics in 3D and fit them with such features as texture, shadows, antialiasing, atmospheric and depth cueing blending or lightening. OpenGL philosophy is to work with 3D transformation matrices and state variables that stay in the same state until a modification occurs. OpenGL is called state machine for this reason. The matrices are setup in a certain order to specify the perspective, point of view, perspective used and deformations to apply to the 3D drawing that are specified after in the code. This feature allows to create a data-structure that can be compiled and treated as a list of primitives that the computer can draw faster and increase the rate of the display output. The drawing primitives can be specified in different ways allowing the user to adapt the code to the algorithm it is associated with, manipulate the language at several degrees of complexity. Pre-processed textures can be distorted and hooked up as paint on the surfaces of the 3D drawing and can be filtered and colored in different ways. Shadows and lightening can be automatically generated by primitives passing the information of position and features of the light(s) as well as the features of the object interacting with this lightening. Antialiasing is the technique that allows to make the lines of a drawing appear sharp and not composed of a serial of points. This requires to blend the color of each points of the line by a factor depending on its distance to the theoretical wanted line. Below is an example of an antialiased line and a non-antialiased one. Other blending functions can be used to produce a fading proportional to the distance from the point of VIEW, producing by the way a visual effect that give to the user a cue on the depth of the scene and allow to reproduce atmospheric effects.

Once the basis most of this language was learned, it was asked to test the new programming skills in upgrading the application already existing. The original application was a composed of several programs written in C with a main program. The compilation is done including several libraries as OpenGL by *#include* macros. The main program is *main* which calls *gl_experiment* which contains the main loop to perform the experiment. This loop calls *sdi* that manage the interaction procedure for the subject and the application.

Other programs contains initialization procedure for the distortion process in *distort,* the initialization for the graphics in *ogl* and additional procedures that contains tools for the computations in *xform.* The definitions of prototype of functions and variables are in *defines.h* and the globals in *global.h*. The compilation implies a *make* UNIX command that use the *Makefile* file included in the current directory. This file involves a special syntax that will not be explained here. A skeleton of a typical *Makefile* has been modified and used for this. To set the environment variables specifying the peripherals used and the location of these over the network to the AD library, a program called *v_setenv* is to source (*source v_setenv).*

The first modification to make to the original application was to implement the possibility to make a variation of the IPD following the same MOCS and staircase distribution employed previously for the depth position. MOCS and staircase distribution method are employed to use a statiscal method to interpret the results. This has been implemented by an additional procedure in making some call to a procedure similar to init_MOCS () in draw_obj.c, and used to initialize the depth positions following the MOCS law.

13. APPLICATION RECONFIGURATION

The application has been reconstructed from the original code and the modules written for each functions of the peripherals. The code also has been divided in several modules assuming the functions of terminal for the user, computation of the distortion, computation of the modeling transformation and rendering of the object. Here is the list of all the modules to assemble:

- main application: initialize environment (variables, devices and processes installation) and enters in the MainLoop controlling and interacting with the following modules and the subject
- user interaction console management module
- viewing transformation and rendering module
- modeling transformation and distortion computation
- translation stepper motor control module
- rotation motor control module
- rotation encoder reading module
- head mounted parallax linear encoder reading module

The original application took too much time to generate the images when the antialiasing and the predistortion computation were enabled and the application was running on an Indigo 2. That is why the application has been designed by modules in order to organize the scheduling of each task in a way that try to make the whole process faster.

The process in charge to interact with the user (not the subject) creates a window that appears in the left top corner of the screen. The parameters of the initialization procedure are displayed and before the user is asked to interact with the keyboard. At this time the MCO is commuted and the user must change the plug 13W3 issued from the console output of the front panel by the one outputted from the MCO. The result is that the top left corner of the original screen is now full screen with the console window into it.

The console during this time has been cleared by the process and the character police has been reduced by four to keep the original size constant. When the Initialisation is finished, the window displays the normal paramaters needed during the experiment that follows. A ‘esc’ hit toggles the current display with a display of all the tunable variables. This variables can be modified since a ‘esc’ produces also an emergency stop of the application.

The generation of graphics implies several steps that are computation of the modeling transformation, the computation of the viewing transformation, the computation of the distortion and finally the rendering. The original application was doing this steps in the given order.

The new application with a parallax resource implies that the viewpoint employed to render the volume can be known only when the subject is in the final position from where he wants to see the object. The viewing transformation computation and the rendering by definition must then occur or re-occur after each move of the subject head, for a same position of the object. However the distortion computation procedure that occured before the viewing transformation originally has been modified to be executed before in order to save time. The distortion is the procedure that moved each point of the generated image to another position in order to catch back the distortion caused by the focus lens front of the television screen. This can be done with the VTX board that our machine have in using texture distortion. The generated image is used as a texture hanged in certain points to stretch it and produce the same results. The distortion computation is necessary since the curvature implied by the focus lens of a theorically straight object can produce errors of position.

The antialiasing is an operation done by hardware if the user enables it; it occurs after the execution of the rendering primitives but before the points are actually rendered on the screens. Because of this it is not possible to reduce the time it takes.

According all the restrictions the time scheduling of the processes has been done as following:

Initialisation

main process
display for user control
(Entry in the main Loop)

Interaction of the application (move the object to the current new position)

main process
display for user control
stepper control module

rotation control module
rotation reading module
distortion and modelling tranformation computation

Intercation subject (move the head mounted parallax and take some views of the object before to answer)

main process
display for user control
head mounted parallax reading
rotation control module
rotation reading module
viewing transformation computation and rendering

Then, these modules where divided in processes following if the task must be done in parallel or not. The first restriction was that the reading module of the head mounted display should restart the viewing and rendering procedure each time a new position of the subject's head was detected. However, the reading process has been done to update the graphics each time the change of position was more than 0.5 mm since the subject could produce new positions just by slightly shaking his hand since the resolution of the linear encoder is 10 micrometer.

To make real time the ‘realtime’ manual page advise to use one microprocessor to answer the realtime calls and one processor for the interruption random in time calls. Also additionnal processes were created if the calls had to be done at the same time on the same processor. The following resulting repartition results of these considerations:

CPU 0	CPU 1
(Initialisation)	
Main process	User control process
(Application interaction)	
Main process	User control process
Modelling transformation computation	rotation control
distortion computation	rotation reading
	stepper control
(Subject interaction)	
Main process	User control process
Viewing transformation computation	rotation control
rendering	rotation reading
	head mounted parallax reading
(loop on application interaction)	

It is tried at this current time to reorganize in such a way the application in order to supervise the time that each CPU is used and may be subject to change if the two processors have not a task load equal. This is the reason why the code of the application is not included in this document. However, this report will be the object of a revision that will show all the results of the final implementation and the modules described above for each function are included in this paper. The current code of the application (main.c, gl_experiment.c ...), the makefile, the executable (user [-options]), the tables are included in the directory ~baillot/public/exp/new of vision at CREOL and measn0-9 at ISIM. Binex copies of this report and the previous report for the old version are included in the concerned directories and called rapport.hqx. These file are Word6.c files.

14. CONCLUSION

In a short time a virtual environment will have been created including a STHMD fitted with a moving and tracking capabilities following one translation and with no rotation. This complete the trainee's knowledge of the basis of virtual reality.

This internship finishes the engineering cursus of the intern. During this internship strong skills have been in the whole process of devellopment of a product. Languages and function of superuser have been done. The internship is going to be continued by a cooperation for the military service to continue to work on the design of the tool. Future work will include measuring the performance of an optical traker recently ordered as well as conducting research on various methods of tracking employed until now.

I would like to thank Jannick Rolland to accept me in her laboratory for this internship and to introduce me to helpfull people, Arthur Kevin and Russel Taylor from UNC for their help to start with the new workstation and reinstall the original application and all the helpfull staff over CREOL and Computer Science at UCF.

Appendices

Following are the appendices giving references and data sheets of the ordered items and scheme of the circuitry of the system. Code is available on the floppy disk in DOS format that comes with the report.